KEY_Practice10_Loops1

December 10, 2021

1 Practice with loops

In this practice, we will review: - how to iterate through a list with a loop - how to use the range function with loops - how to create a nested loop

As you have seen, loops give us the ability to loop through, or *iterate* through, a list of items and perform the same action with each item in the list. A for loop follows this pattern:

```
for [item] in [list]
  [do something with item]
```

Some other things to remember before practicing: - The `range` function gives you a list of numbers between two parameters. For example, `range(0,3)` would give you the list `[0,1,2]`. It is *inclusive* of the first parameter, but *exclusive* of the second parameter. - You can put for loops inside each other, but it is **very** important to make sure both loops have different variable names. For example, this is okay:

```
for num1 in list_1:
  for num2 in list_2:
    print(num1 + num2)
```

but this is not:

```
for num in list_1:
  for num in list_2:
    print(num + num)
```

The above code will probably run, but it won't give you the output you expect!

In this lesson, we are going to work with information about the planets!

[19]:
```
# create a list of planets
planets = ['mercury', 'venus', 'earth', 'mars', 'jupiter', 'saturn', 'uranus',␣
 ↪'neptune', 'pluto']
```

First, let's create a string containing the names of all the planets. We want something that looks like this:

"mercury venus earth mars jupiter saturn uranus neptune pluto"

This can be done by creating an empty string, then adding the name of each planet using a for loop!

[21]:
```
# create an empty string called planet_string (hint: an empty string is just␣
 ↪opening & closing quotation marks)
planet_string = ''

# use a for loop to add each planet's name to the string
# remember that each time we add a planet we also want to add a space
for planet in planets:
    planet_string += planet + ' '

# print the value of planet_string
print(planet_string)
```

```
mercury venus earth mars jupiter saturn uranus neptune pluto
```

What if we wanted to create a string that only contained the planets further from the sun than Earth (these are the planets that come after Earth in our list).

[11]:
```
# create an empty string called after_earth
after_earth = ''

# create a for loop similar to above, but only iterating over the planets after␣
 ↪earth in the list
for index in range(3, len(planets)):
    after_earth += planets[index] + ' '

# print the value of after_earth
print(after_earth)
```

```
mars jupiter saturn uranus neptune pluto
```

What if we also wanted to look at more data about the planets? We can create a 2D list that contains more information about each planet.

[8]:
```
# the number of moons each planet has
moons = [0,0,1,2,67,62,27,14,5]

# the distance of each planet from the sun (in millions of miles)
distances = [35.98,67.24,92.96,141.6,483.8,890.8,1784,2793,3670]

# create a 2D list (list of lists) made up of the planets, moons,
# and distances lists & assign it to the variable 'space'
space = [planets, moons, distances]
```

We can visualize our 2D list as looking something kind of like this:

```
mercury | 0 | 35.98
venus   | 0 | 67.24
earth   | 1 | 92.96
mars    | 2 | 141.6
```

```
...   |...|   ...
```

But when we print our 2D list, that isn't what it looks like

[9]:
```
# print the 2D list space
print(space)
```

```
[['mercury', 'venus', 'earth', 'mars', 'jupiter', 'saturn', 'uranus', 'neptune',
'pluto'], [0, 0, 1, 2, 67, 62, 27, 14, 5], [35.98, 67.24, 92.96, 141.6, 483.8,
890.8, 1784, 2793, 3670]]
```

Fortunately, we now know how to use nested for loops! Using one for loop to iterate over the "columns" of the 2D list and another to iterate over the "rows" of the 2D list, we can print out our 2D list so that it looks like a table.

First we will loop through the planets and make an empty string that will become each row in the table. Then for each empty string/row, we will add the name of the planet, number of moons, and distance from the sun.

[16]:
```
# first loop through the planets
for planet_index in range(0, len(planets)):
    # make an empty string to construct each row
    row_string = ''
    # make another loop here to get the right entries from the planets, moons,␣
 ↪and distances list
    # (hint: loop through "space" as we did with "planets" above)
    # (hint 2: add str(space[your_loop_index][planet_index]) plus the tab␣
 ↪character "\t" to row_string)
    for column_index in range(0, len(space)):
        row_string += str(space[column_index][planet_index]) + '\t'
    print(row_string)
```

```
mercury 0       35.98
venus   0       67.24
earth   1       92.96
mars    2       141.6
jupiter 67      483.8
saturn  62      890.8
uranus  27      1784
neptune 14      2793
pluto   5       3670
```

[ ]:
```

```